IE 3186: Approximate Dynamic Programming **Fall 2018**

Lecture 4: Asynchronous VI and Bellman Reformulations

Lecturer: Daniel Jiang *Scribes: Mingyuan Xu*

References:

D.P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, Vol. 2, 4th ed, Athena Scientific, Belmont MA, 2012. (§2.6)

W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed, Wiley & Sons, 2007. (§4.6)

4.1 General Asychronous Model

We are trying to solve fixed point equation: $J = TJ$. A general asynchronous model is as follows:

- One "processor" is responsible for each state (or component) of $J = (J_1, J_2, \ldots J_n)$.
- Let $R_l = \{$set of iterations at which processor l updates state $l\}$.
  - This model can also model the more realistic case where there are fewer processors than states. Can simply add virtual processors that mimic a physical processor that updates multiple states.
- States are left unchanged if they are not updated.
- J_l^t is the approximation of J_l^* at iteration t.
- When processor l updates state l at some iteration $t \in R_l$, it sees delayed versions of J_j for $j \neq l$. The communication delay from processor $j \to l$ is

  $$t - \tau_{(l,j)}(t),$$

  meaning that processor l sees $J_j^{\tau_{(l,j)}(t)}$. The update equation is:

  $$J_l^{t+1} = \begin{cases} T(J_1^{\tau_{(l,1)}(t)}, J_2^{\tau_{(l,2)}(t)}, \ldots, J_n^{\tau_{(l,n)}(t)})(l) & \text{if } t \in \mathcal{R}_l \\ J_l^t & \text{if } t \notin \mathcal{R}_l. \end{cases} \tag{4.1}$$

- The standard case of no-delay is $\tau_{(l,j)}(t) = t$.
- The asynchronous VI algorithm can be written as a special case of this model:

$$J_l^{t+1} = \begin{cases} T(J_1^t, J_2^t, \ldots, J_n^t)(l) & \text{if } l = x_t \\ J_l^t & \text{if } l \neq x_t \end{cases}$$

where $T(J_1^t, J_2^t, \ldots, J_n^t)(l)$ denotes the l^{th} component of $T(J_1^t, J_2^t, \ldots, J_n^t) = TJ^t$.

Assumption 4.1. *States are visited infinitely often, i.e., $|\mathcal{R}_l| = \infty$ for each state l.*

Assumption 4.2. *Information is always renewed or "caught up," i.e., $\lim_{t\to\infty} \tau_{(l,j)}(t) = \infty$ for every pair of processors (l, j).*

Proposition 4.3 (Convergence)**.** *Under previous assumptions, the asynchronous algorithm given in (4.1) converges to J^*, the fixed point of $J = TJ$: $J^t \to J^*$.*

Proof. Let $S(k) = \{J : \|J - J^*\|_\infty \leq \gamma^k \|J^0 - J^*\|_\infty\}$, so we have a shrinking set of boxes: $S(k+1) \subseteq S(k)$ for each k.

- By the contraction property of T, if $J \in S(k)$, then $TJ \in S(k+1)$.
- Note that $S(k) = S_1(k) \times S_2(k) \times \cdots \times S_n(k)$, where $S_l(k)$ is an interval in the l^{th} dimension.
- The boxes shrink to a single point J^*.

Main idea of the analysis:

1. Suppose $J \in S(k)$, then if we update J_l by applying T to J and keep the l^{th} component, then the new value function J' is still $\in S(k)$.
2. Once $J_l \in S_l(k)$ and ignoring delays, then J_l will enter $S_l(k+1)$ the first time it is updated after entire vector J enters $S(k)$.
3. So the iterates progressively enter smaller and smaller boxes and eventually converge to J^*.

Prove by induction: For each $k \geq 0$, show that there exists time t_k, such that

- I_1: $J^t \in S(k)$ for all $t \geq t_k$.
- I_2: $(J_1^{\tau_{(l,1)}(t)}, J_2^{\tau_{(l,2)}(t)}, \ldots, J_n^{\tau_{(l,n)}(t)}) \in S(k)$ for all l, $t \in \mathcal{R}_l$, and $t \geq t_k$.

That can be interpreted to mean:

1. The estimate is in $S(k)$ once t is sufficiently large.
2. All delayed versions of the estimates are also in $S(k)$.

Base case ($k = 0$) is true because $J^0 \in S(0)$ by definition.

Now assume for k, the two induction hypotheses (I_1, I_2) are true. We now try to prove that there exists t_{k+1} such that (I_1, I_2) hold.

Let $t(l)$ be the first time l is updated after t_k: $t(l) = \min\{i : i \in R_l, t \geq t_k\}$.

By the contraction property of Bellman operator T, we know that once delayed versions enter $S(k)$ after t_k (I_2), the update equation for $J_l^{t(l)+1} \in S_l(k+1)$:

$$J_l^{t(l)+1} = \begin{cases} T(J_1^{\tau_{(l,1)}(t)}, J_2^{\tau_{(l,2)}(t)}, \ldots, J_n^{\tau_{(l,n)}(t)})(l) & \text{if } t \in R_l \\ J_l^t & \text{if } t \notin R_l \end{cases}$$

sends $J_l^{t(l)+1}$ into $S_l(k+1)$.

In fact, for all $t \in R_l$ and $t \geq t(l)$, it holds that $J_l^{t+1} \in S_l(k+1)$. Since J_l^t is not changed between updates, we actually know that

$$J_l^t \in S_l(k+1) \quad \text{for all} \quad t \geq t(l) + 1. \tag{4.2}$$

Let $t'_k = \max_l \{t(l)\} + 1$, the time after which all processors l have updated their respective component l at least once after t_k. Since (4.2) holds for all processors l, this means that:

$$J^t = (J_1^t, J_2^t, \ldots, J_n^t) \in S(k+1) \quad \text{for all} \quad t \geq t'_k.$$

The final step is to select t_{k+1} such that all delayed versions are also in $S(k+1)$. We can simply "wait" until this is true. By assumption, $\tau_{(l,j)}(t) \to \infty$ as $t \to \infty$. Choose t_{k+1} sufficiently large such that $\tau_{(l,j)}(t) \geq t'_k$ for all l, j when $t \geq t_{k+1}$.

Thus, the time index t_{k+1} verifies both induction hypotheses. □

Remark 4.4. *This ADP algorithm is "approximate" only in the sense that we compute Bellman in an asychronous manner (not all states). However, for each Bellman update, we compute it* ***exactly*** *for the states that we visit. Can we consider approaches where there is noise in the Bellman computation as well?*

4.2 Q-factor Reformulation

Suppose we sample $\hat{w}$ from the same distribution of w, then we could try a sampled version of the Bellman update:

$$(TJ)(i) \approx \min_u \; [(g(i, u, \hat{w}) + \gamma J(f(i, u, \hat{w})] =: (\hat{T}J)(i, \hat{w}).$$

But $TJ(i) \geq \mathbf{E}_{\hat{w}}[(T\hat{(}J)(i,\hat{w})]$. It is hard to imagine that a biased observation of TJ can lead to good algorithms.

We would prefer $\hat{J}$ be an unbiased observation of TJ, i.e.,

$$(\hat{T}J)(i) = (TJ)(i) + \epsilon, \ \mathbf{E}(\epsilon) = 0.$$

Goal: let's try to reformulate the Bellman equation so that unbiased estimates are easy. If successful, then we will have a new Bellman operator T' such that it is easy to observe $(T'J)(i) + \epsilon$.

Definition 4.5. *Suppose $U(i) = U$ for all i (for simplicity). Define the Q-factor or (state-action value function):*

$$Q^*(i,u) = \mathbf{E}\left[g(i,u,w) + \gamma J^*(f(i,u,w))\right] = g(i,u) + \gamma \mathbf{E}\left[J^*(f(i,u,w))\right]$$

where action u is taken out of state i for one step and then π^* followed afterwards.

Then, we have

$$J^*(i) = \min_u Q^*(i,u), \quad \pi^*(i) \in \operatorname{argmin} Q^*(i,u).$$

A nice observation is that if we have Q^*, then there is no need to compute an expectation when implementing π^*.

Note that we can also write versions for the fixed policy case:

$$Q^\pi(i,u) = \mathbf{E}\left[g(i,u,w) + \gamma J^\pi(f(i,u,w))\right] = g(i,u) + \gamma \mathbf{E}\left[J^*(f(i,u,w))\right].$$

The Q-factor version of Bellman equation is

$$Q^*(i_k,u_k) = g(i_k,u_k) + \gamma \mathbf{E}\left[\min_{u_{k+1}} Q(f(i_k,u_k,w_{k+1}),u_{k+1})\right].$$

Theoretical properties from T essentially all follow using the same proof ideas. We can aim to find Q^* instead of J^*.

Algorithm 4.6. *Q-factor Value Iteration:*

(1) Choose Q_0 arbitrarily.

(2) Iterate $Q_{k+1} = FQ_k$, where $(FQ)(i,u) = g(i,u) + \gamma \mathbf{E}\left[\min_{u'} Q^*(f(i,u,w),u')\right]$.

Then, $Q_k \to Q^*$.

We now have have an easy way to get an unbiased estimate because min and expectation are interchanged:

$$(\hat{F}Q)(i,u,\hat{w}) = g(i,u) + \min_{u'} Q(f(i,u,\hat{w}),u'),$$

$$(FQ)(i,u) = \mathbf{E}[(\hat{F}Q)(i,u,\hat{w})].$$

Unfortunately, the new "state space" is $|U|$ times larger than before.

4.3 Post-decision Reformulation

Question: is a simpler variable that contains the same information as (i, u)? The *post-decision state* is the state of the system after u is decided, but before w is revealed. The original transition f is:

$$i_{k+1} = f(i_k, u_k, w_{k+1})$$

Consider the following decomposition of transition f:

$$i_k^u = f_1(i_k, u_k), \; i_{k+1} = f_2(i_k^u, w_{k+1}).$$

Then, we have:

$$\begin{aligned} J^*(i) &= \min_u \; g(i,u) + \gamma \mathbf{E}\left[J^*(f(i,u,w_{k+1}))\right] \\ &= \min_u \; g(i,u) + \gamma \underbrace{\mathbf{E}\left[J^*(f_2(i_k^u, w_{k+1}))\right]}_{\tilde{J}(i_k^u)} \\ &= \min_u \; g(i,u) + \gamma \tilde{J}(f_1(i_k, u_k)). \end{aligned}$$

Therefore, the post-decision Bellman equation is given by:

$$\tilde{J}(i_k^u) = \mathbf{E}\Big[\min_{u_{k+1}} g(i_{k+1}, u_{k+1}) + \gamma \tilde{J}(f_1(i_{k+1}, u_{k+1})\Big]$$

Example 4.7 (Inventory Control)**.** *Consider the following MDP model for the standard inventory control problem:*

- *Variables:*
  - *State x_k: current stock,*
  - *Decision $u_k \geq 0$: how much to order,*
  - *Noise D_{k+1} is the demand (assume it is independent over time periods).*
- *Transition function: $x_{k+1} = x_k + w_k - D_{k+1}$*
- *Cost function:*

$$\begin{aligned} g(x_k, u_k, D_{k+1}) &= cu_k + h(x_k + u_k - D_{k+1})^+ + b(D_{k+1} - x_k - u_k)^+, \\ \bar{g}(x_k, u_k) &= \mathbf{E}\big[g(x_k, u_k, D_{k+1})\big]. \end{aligned}$$

Different formulations:

(a) $J^*(x_k) = \min_{u_k} \bar{g}(x_k, u_k) + \gamma \mathbf{E}[J^*(x_{k+1})]$.
State: 1*-dim;* $\min \mathbf{E}$ *formulation.*

(b) $Q^*(x_k,u_k) = \bar{g}(x_k,u_k) + \gamma \mathbf{E}\big[\min_{u_{k+1}} Q^*(x_{k+1},u_{k+1})\big]$.
State: 2-dim; $\mathbf{E}\min$ *formulation.*

(c) *Let* $y_k = x_k + u_k =: f_1(x_k,u_k)$ *and* $x_{k+1} = y_k - D_{k+1} =: f_2(y_k, D_{k+1})$.
$\tilde{J}(y_k) = \mathbf{E}[J^*(y_k - D_{k+1})] = \mathbf{E}\Big[\min_{u_{k+1}} \bar{g}(x_{k+1},u_{k+1}) + \gamma \tilde{J}(y_{k+1})\Big]$
State: 1-dim; $\mathbf{E}\min$ *formulation. Ideal case.*

Example 4.8. *(Multiple Locations/Prices)*

Example 4.7 except the manager can order from n locations/suppliers at fluctuating cost $\{c_k^1, c_k^2, \ldots, c_k^n\}$. Let x_k be the current stock.

- *Variables:*
  - *State Space:* $(x_k, c_k^1, c_k^2, \ldots, c_k^n)$,
  - *Decision Space:* $0 \le u_k^i \le u_{\max}^i$ *for* $k = 1, 2, \ldots, n$,
  - *Noise D_{k+1} is the demand (assume it is independent over time periods).*
- *Transition function:* $r_{k+1} = r_k + \sum_i u_k^i - D_{k+1}$, c_{k+1}^i *drawn from a distribution.*
- *Cost-to-go function:*
  $g(r_k, (c_k^i), (u_k^i), D_{k+1}) = \sum_i c_k^i u_k^i + h(r_k + \sum_i u_k^i - D_{k+1})^+ + b(D_{k+1} - r_k - \sum_i u_k^i)^+$,
  $\bar{g}(r_k, c_k^i, u_k^i) = \mathbf{E}[g(r_k, c_k^i, u_k^i, D_{k+1})]$.

Different formulations:

(a) *Standard formulation has* $(n+1)$*-dim state and* $\min \mathbf{E}$.

(b) *Q-factor formulation has* $(2n+1)$*-dim state and* $\mathbf{E}\min$.

(c) *Let* $y_k = r_k + \sum_i u_k^i$,
$\tilde{J}(y_k) = \mathbf{E}[J^*(y_k - D_{k+1}, (c_{k+1}^i))] = \mathbf{E}\Big[\min_{(u_{k+1}^i)} \bar{g}(r_{k+1}, c_{k+1}^i, u_{k+1}^i) + \gamma \tilde{J}(y_{k+1})\Big]$.
Post-decision formulation only has 1-dim state and $\mathbf{E}\min$*!*